



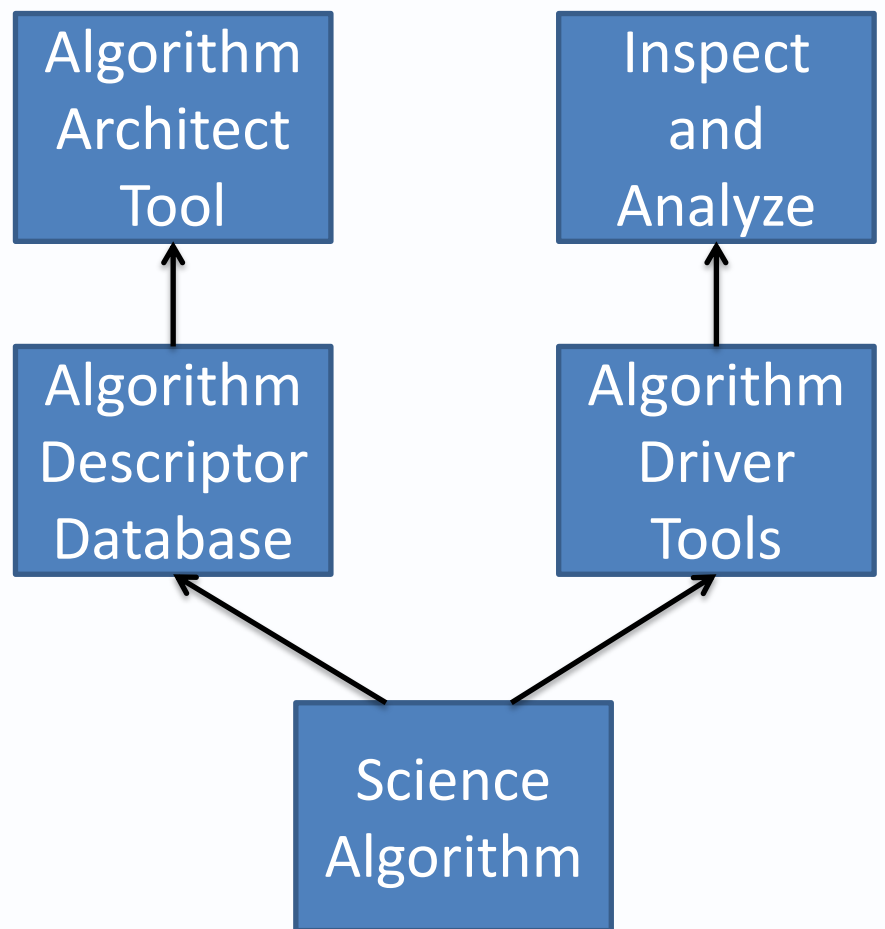
# The Algorithm Workbench

Data-Driven Software for Ground Processing System Development, Test, and Operations

Alexander Werbos, David Hogan, Daniel Hunt, Erik Steinfeld, and T. Scott Zaccheo  
Atmospheric and Environmental Research, Inc., Lexington, MA  
<http://www.aer.com/>

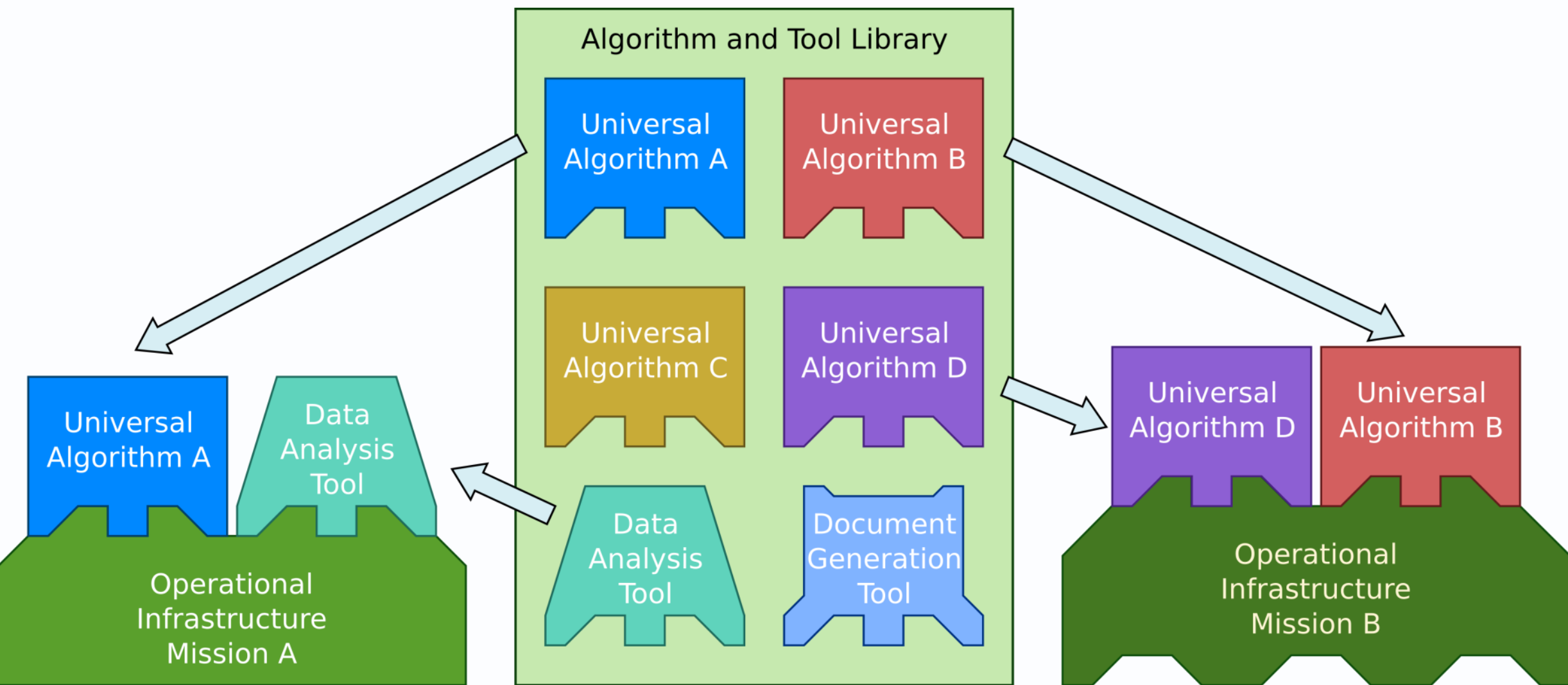
## Introduction

This work describes the **Algorithm Workbench**, a set of software interfaces and tools that allow users to effectively create, document, execute, and update software data processing systems. This software is specifically designed to reduce the overhead involved in research to operations, and enable direct sharing of science algorithms across development, testing and production environments. Each component, shown below, is part of a complete data processing system, which is designed to be adaptable and customizable to user needs in both operational and development contexts.



## The Vision: A Common Algorithm Library

The Algorithm Workbench is designed from the ground up to be a component-based system that can be adapted to user needs. The various subsystems are built around open interfaces to allow substitution or alteration of any individual component without losing the benefits of all the others. The ultimate goal of the package is to provide the user with the ability to use common libraries of algorithms and tools that can be easily and quickly adapted for use in new systems. This allows new missions to leverage the effort expended in the creation, operationalization, and maintenance of previous algorithms by ensuring that the same code base can be used in multiple environments, independent of the infrastructure required and other algorithms used. The ADDB will automatically determine the necessary inputs, and where each algorithm will fit in the mission-specific processing tree. And while these algorithm and tool libraries can be centralized, the fragment-based nature of the ADDB allows users to easily manage their own libraries, and merge the algorithms and data with those from other sources. These can be layered with project-specific tools, drawing from the data contained within the ADDB, to produce the required documentation and description artifacts, dramatically reducing the workload required to create and manage those deliverables. These features allow for a user-friendly, adaptable, and efficient approach to the creation of use of new remote sensing data processing systems.



## Algorithm Descriptor Database (ADDB)

- Programmatically-Accessible database that stores information about algorithms and datatypes
  - All algorithm inputs and outputs are stored in abstracted form, allowing the end-user to assign physical types and grids
- Allow system development driven by automated tools rather than manual management
  - Automated analysis can describe and verify all inputs and data flows, avoiding the problem of a missed data product amongst the hundreds involved in an operational system
  - Document generation tools can read ADDB information to create data dictionaries, component listings, etc. By generating these mechanically, overhead and error involved in manual management are reduced.
- Fragment-based architecture for ease of distribution
  - ADDB information is packaged in a fragment file with the associated algorithm code.
  - Individual algorithms can be selected for use without requiring dependencies on unrelated ones.
  - No central repository to be distributed and managed

# Using the Algorithm Workbench to drive GOES-R Operational Algorithms on Himawari AHI data

## Adapting GOES-R Algorithms to Himawari Data

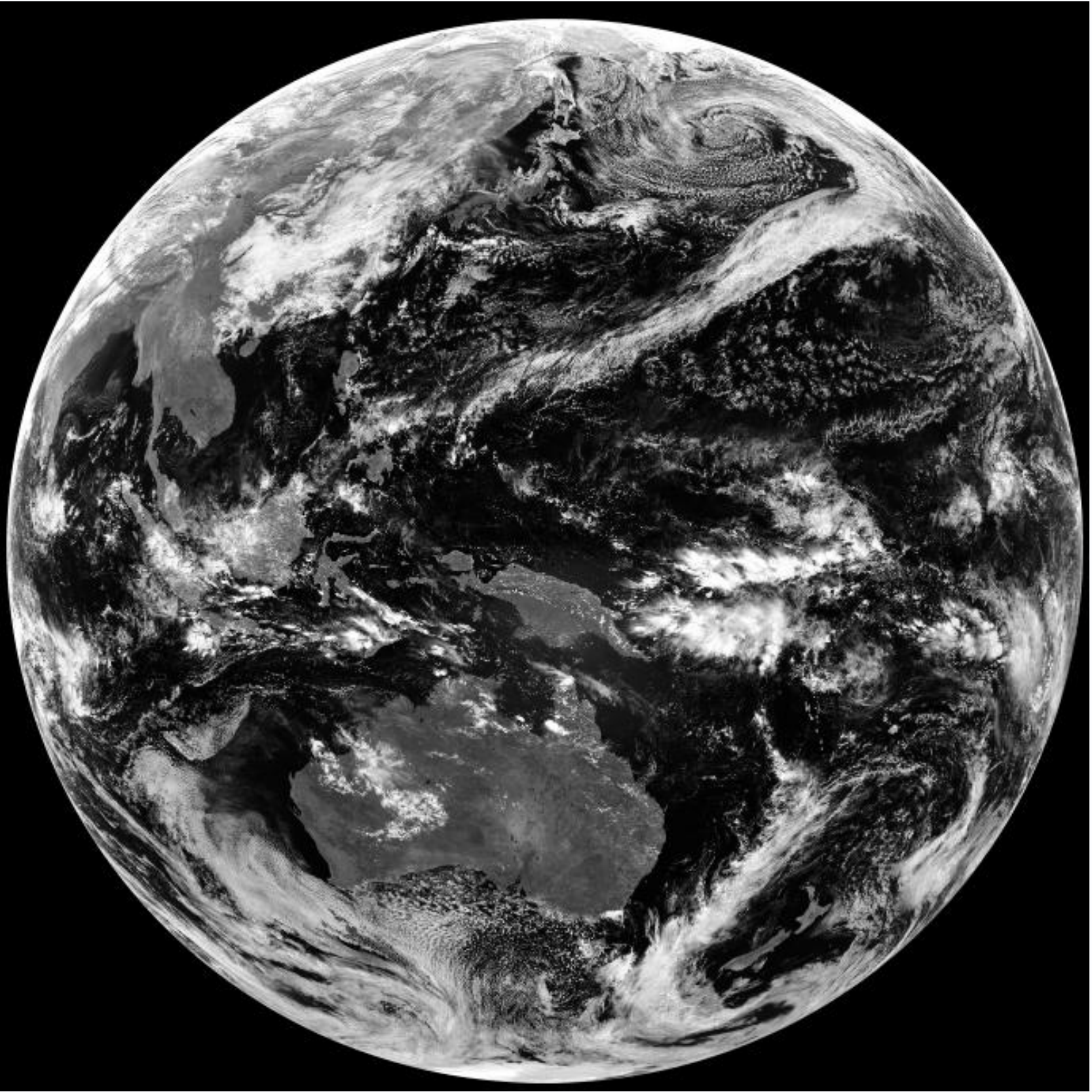
Using the AER algorithm workbench, the operational GOES-R algorithms have been executed, without any code changes, on newly-released Himawari AHI data. By maintaining the same code base across multiple missions, the latest updates can be shared, and users can be provided consistent product outputs across the globe.

To prepare the algorithms for run, a simple mapping was created from the AHI (Himawari) channels to ABI (GOES-R) channels, and provided as an input to the algorithm workbench. In addition, semi-static and ancillary inputs (NWP, Reynolds global SST, snow masks, etc.) were prepared for the associated observation times and sub-satellite position.

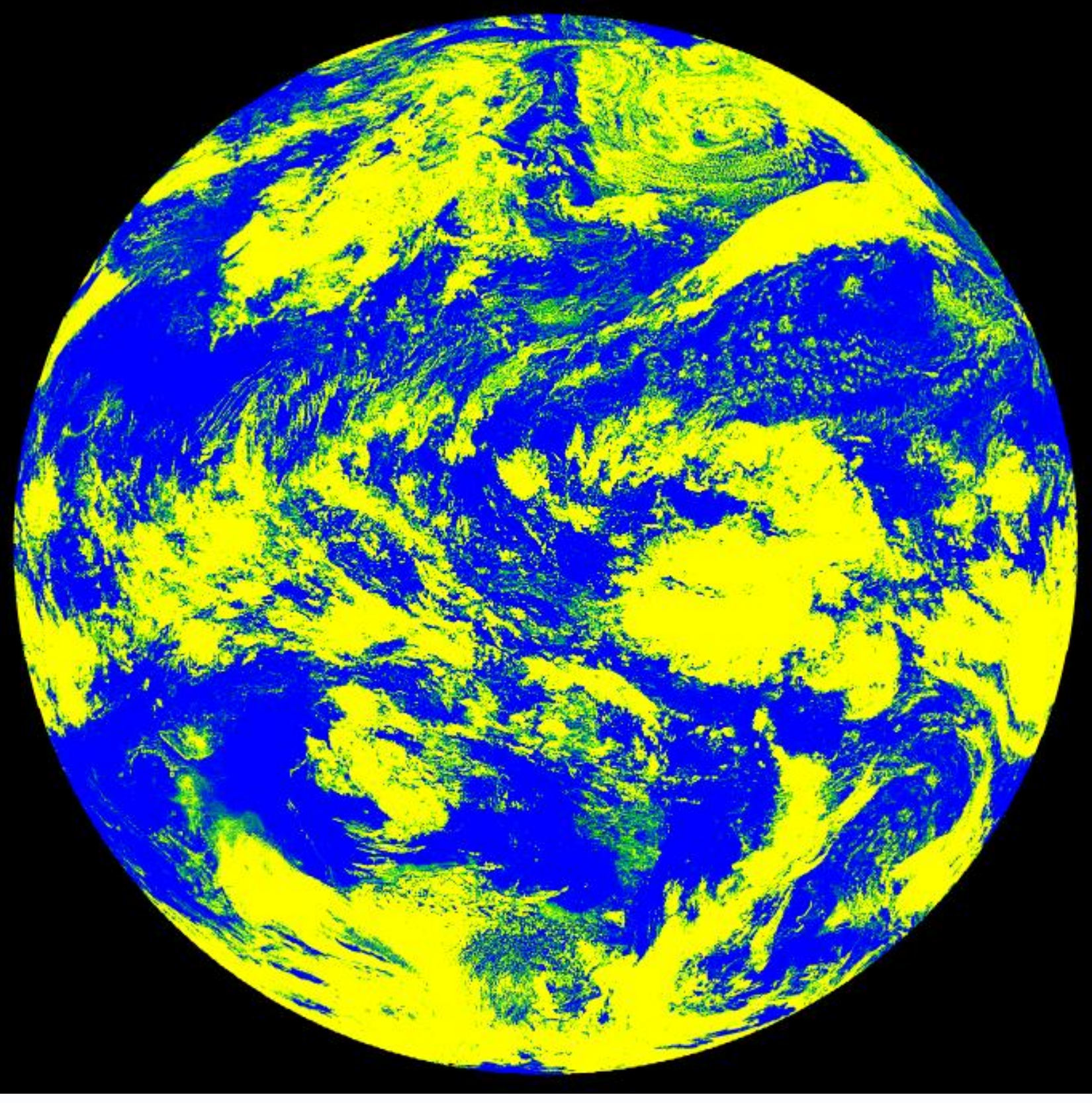
In this scenario, we ran the core cloud algorithms, Cloud Mask, Cloud Type and Phase, and Cloud Top Height and Temperature. The results are shown below.

		Approximate
Himawari AHI	GOES-R ABI	Wavelength (μm)
Band1	Band1	0.47
Band2	No Match	0.51
Band3	Band2	0.64
Band4	Band3	0.86
No Match	Band4	1.38
Band5	Band5	1.61
Band6	Band6	2.25
Band7	Band7	3.90
Band8	Band8	6.19
Band9	Band9	6.95
Band10	Band10	7.34
Band11	Band11	8.50
Band12	Band12	9.61
Band13	Band13	10.35
Band14	Band14	11.20
Band15	Band15	12.30
Band16	Band16	13.30

## Himawari Band 4 (0.86 μm) Reflectances

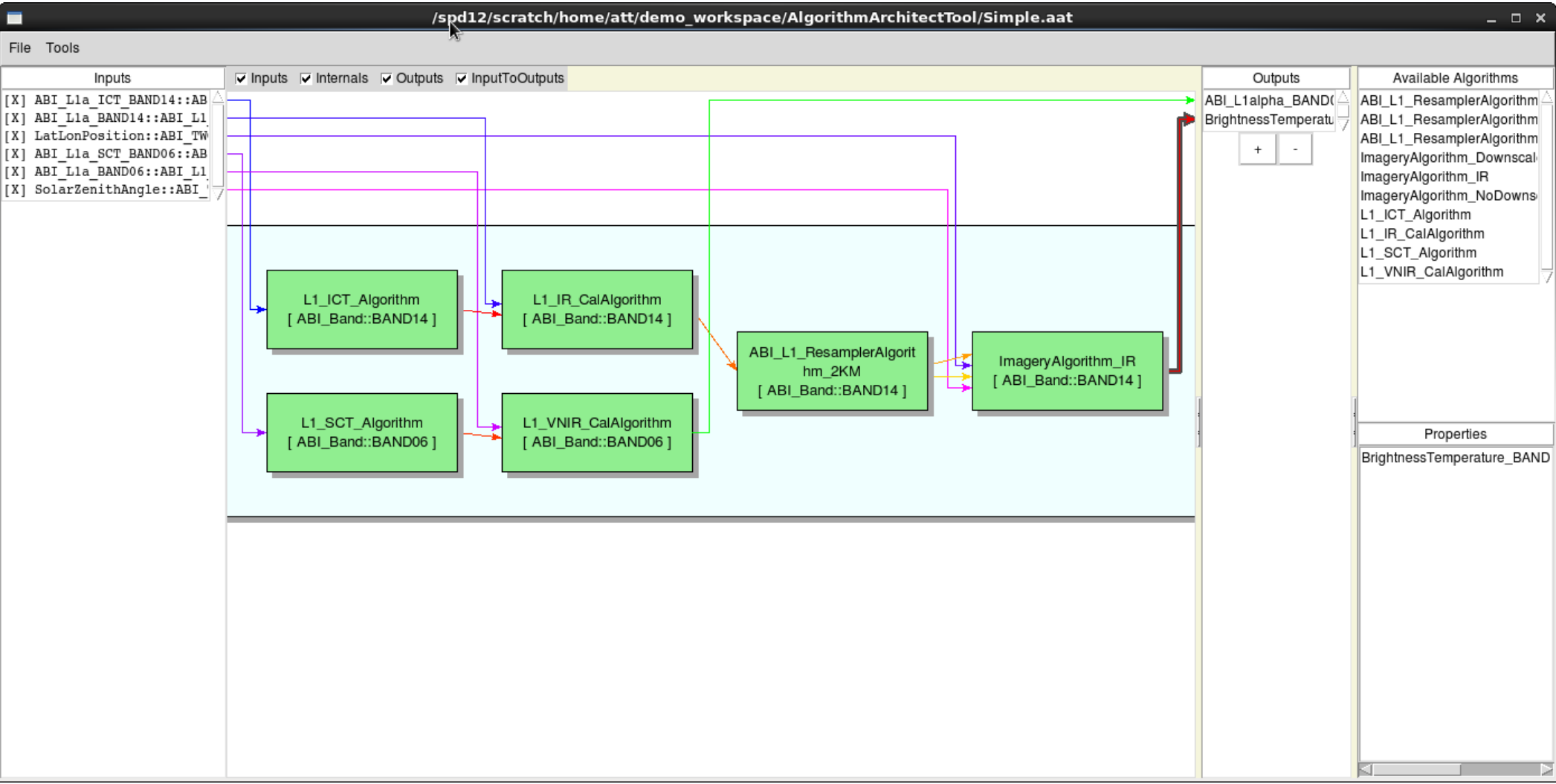


## Cloud Mask



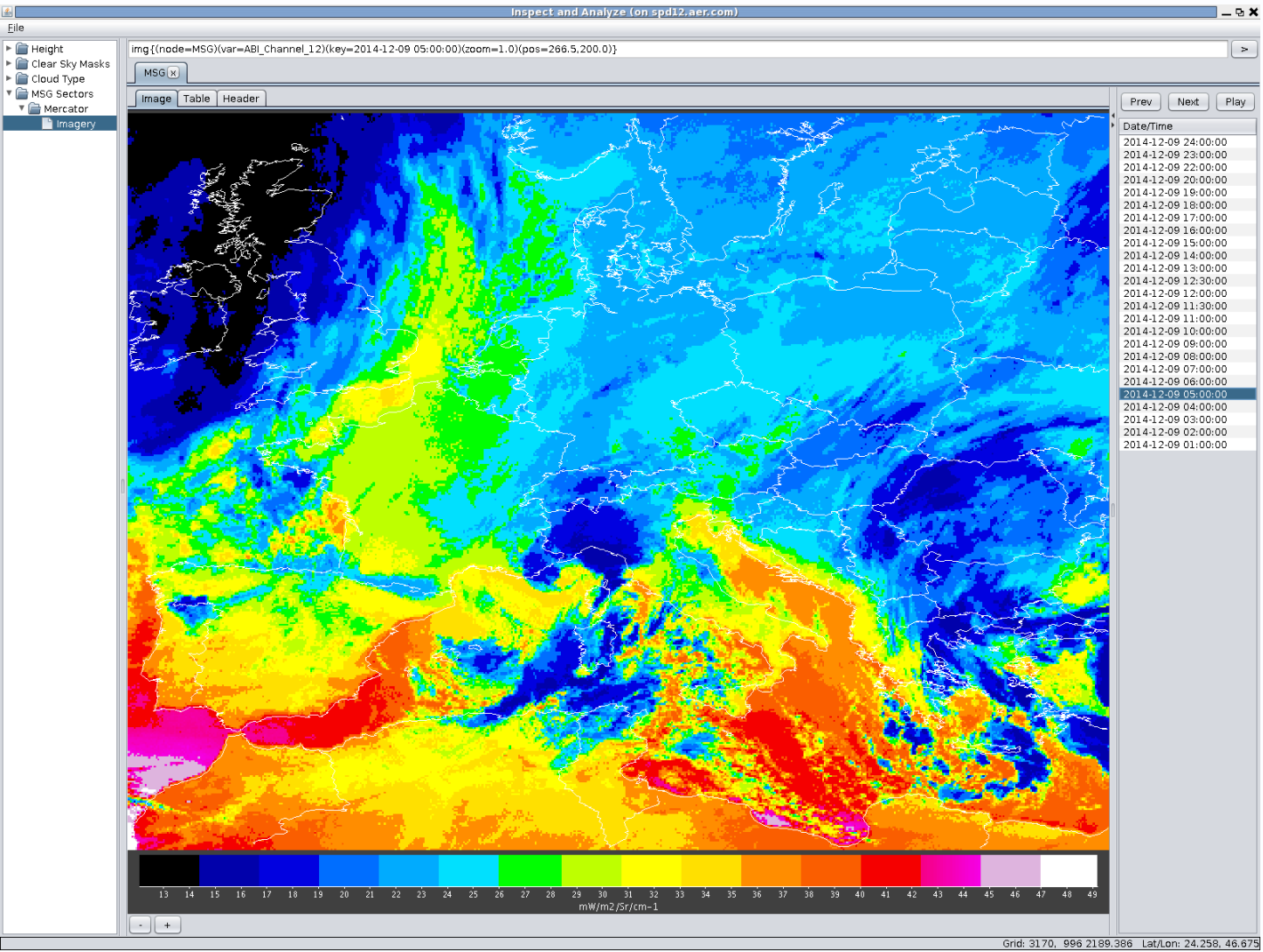
## Algorithm Architect Tool

- User can select desired outputs and automatically generate algorithm trees to produce the desired outputs
- Allows user to select input data files and directly run the displayed tree using the Algorithm Driver
- Exploration features give users the ability to trace dataflows, examine algorithms, and validate that all inputs are present and correct

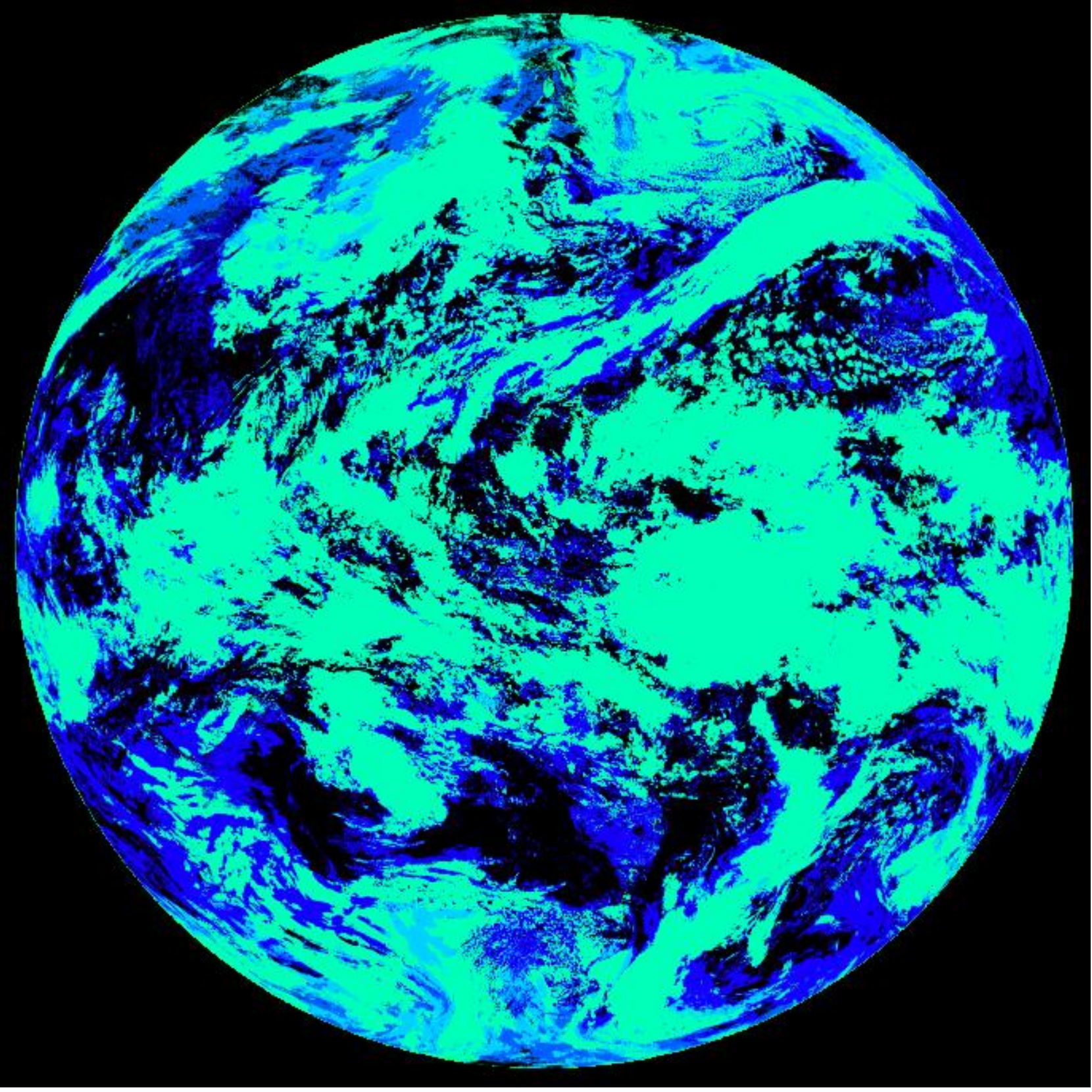


## Inspect and Analyze

- Java-based tool with operational legacy: previous version used to monitor product quality in GOES-R ground segment
- Offers automated re-projection between grids, pan, zoom, and sub-scene sharing
- Capable of automatically updating as new data arrive, allowing users to monitor live processing
- Layered software architecture allows inputs to be gathered from a variety of formats (NetCDF, HDF, etc.) and multiple different infrastructures (file system, database)



## Cloud Phase



## Cloud Top Temperature

